Weighted Model Counting Without Parameter Variables

14th February 2021

1 Introduction

Notation. For any propositional formula ϕ and $p, q \in \mathbb{R}_{\geq 0}$, let $[\phi]^p_q \colon 2^X \to \mathbb{R}$ be a pseudo-Boolean function defined as

$$[\phi]^p_q \coloneqq \begin{cases} p & \text{if } \phi \\ q & \text{otherwise.} \end{cases}$$

References.

- Related work without publicly available implementations:
  - direct compilation to SDDs [2]
  - direct compilation to PSDDs, also eliminating parameter variables (a thesis)
  - maybe two more papers

Notes.

- Apparently, the DPMC paper already shows that taking the first offered decomposition tree is best.
- It is already well-known that WMC is FPT.

2 Parameter Variable Elimination

Notes.

- Let X_P be the set of parameter variable and X_I be the set of indicator variables.
- Parameter variables are either taken from the LMAP file (for encodings produced by Ace) or assumed to be the variables that have both weights equal to 1.
- If a parameter variable in a clause is 'negated', we can ignore the clause. We assume that there are no clauses with more than one instance of parameter variables.
- The second **foreach** loop can be performed in constant time by representing ϕ' as a list and assuming that the two 'clauses' are adjacent in that list (and incorporating it into the first loop).
- The d map is constructed in $\mathcal{O}(|X_P| \log |X_P|)$ time (we want to use a data structure based on binary search trees rather than hashing).
- `rename` can be implemented in $\mathcal{O}(\log |X_P|)$ time.
- This may look like preprocessing, but all the transformations are local and thus can be incorporated into an encoding algorithm with no slowdown. In fact, if anything, the resulting algorithm would be slightly faster, as it would have less data to output.

Algorithm 1: WMC instance transformation

Data: an (old-format) WMC instance (ϕ, X_I, X_P, W)
Result: a (new-format) WMC instance (ϕ', ω)
$\phi' \leftarrow \emptyset$;
$\omega \leftarrow 1$;
let $d \colon X_P \to \mathbb{N}$ be defined as $v \mapsto |\{u \in X_P \mid u \leq v\}|$;
foreach *clause* $c \in \phi$ **do**
    if $c \cap X_P = \{v\}$ *for some* v **and** $W(v) \neq 1$ **then**
        if $|c| = 1$ **then**
            $\omega \leftarrow \omega \times W(v)$;
        else
            $\phi' \leftarrow \phi' \cup \left\{ \left[\bigwedge_{l \in c \setminus \{v\}} \neg l \right]_1^{W(v)} \right\}$;
    else if $\{v \mid \neg v \in c\} \cap X_P = \emptyset$ **then**
        $\phi' \leftarrow \phi' \cup \{[c]_0^1\}$;
foreach *indicator variable* $v \in X_I$ **do**
    if $\{[v]_1^p, [\neg v]_1^q\} \subseteq \phi'$ *for some* p *and* q **then**
        $\phi' \leftarrow \phi' \setminus \{[v]_1^p, [\neg v]_1^q\} \cup \{[v]_q^p\}$;
replace every variable v in ϕ' with `rename(`v`)`;
return (ϕ', ω);
Function `rename(`v`)`**:**
    $S \leftarrow \{u \in X_P \mid u \leq v\}$;
    if $S = \emptyset$ **then return** v;
    return $v - d(\max S)$;

3 Parameterised Complexity of DPMC

Notes.

- Summary of results
  - We establish DPMC inference as fixed-parameter tractable.
  - We experimentally show that cw+DPMC is best on low-to-moderate treewidth instances, and cd06+c2d overtakes cw+DPMC on higher treewidth instances.
- By DPMC, we always mean DMC+lg.

TODO: Define:

- Formal definition of a previous WMC instance (CNF, literal weight function) and the new definition (set of $2^X \to \mathbb{R}_{\geq 0}$ pseudo-Boolean functions and a constant). Note that the constant idea is borrowed from `bklm16`.
- Boolean formula in CNF (perhaps this is too trivial to define),
- primal graph of a CNF formula (a.k.a. Gaifman/(variable) interaction/connectivity/clique/representing graph),
- Bayesian network (I already have a definition of these last two),
- moralisation of a Bayesian network (or of any DAG),

Theorem 1 ([5], rephrased)**.** *BN Inference (for all algorithms that accept arbitrary instances) has a lower bound that's linear in the size of the BN and exponential in the treewidth of its moralisation (provide the exact formula).*

Definition 1 ([4], verbatim)**.** Let X be a set of Boolean variables and ϕ be a CNF formula over X. A *project-join tree* (PJT) of ϕ is a tuple (T, r, γ, π) where:

- T is a tree with root $r \in \mathcal{V}(T)$,
- $\gamma \colon \mathcal{L}(T) \to \phi$ is a bijection between the leaves of T and the clauses of ϕ, and
- $\pi \colon \mathcal{V}(T) \setminus \mathcal{L}(T) \to 2^X$ is a labelling function on internal nodes.

Moreover, (T, r, γ, π) must satisfy the following two properties:

1. $\{\pi(n) : n \in \mathcal{V}(T) \setminus \mathcal{L}(T)\}$ is a partition of X, and
2. for each internal node $n \in \mathcal{V}(T) \setminus \mathcal{L}(T)$, variable $x \in \pi(n)$, and clause $c \in \phi$ such that x appears in c, the leaf node $\gamma^{-1}(c)$ must be a descendant of n in T.

Definition 2 ([4])**.**

$$\texttt{Vars}(n) \coloneqq \begin{cases} \texttt{Vars}(\gamma(n)) & \text{if } n \in \mathcal{L}(T) \\ \left(\bigcup_{o \in \mathcal{C}(n)} \texttt{Vars}(o)\right) \setminus \pi(n) & \text{if } n \notin \mathcal{L}(T). \end{cases}$$

$$\texttt{size}(n) \coloneqq \begin{cases} |\texttt{Vars}(n)| & \text{if } n \in \mathcal{L}(T) \\ |\texttt{Vars}(n) \cup \pi(n)| & \text{if } n \notin \mathcal{L}(T). \end{cases}$$

The *width* of a PJT (T, r, γ, π) is $\texttt{width}(T) \coloneqq \max_{n \in \mathcal{V}(T)} \texttt{size}(n)$.

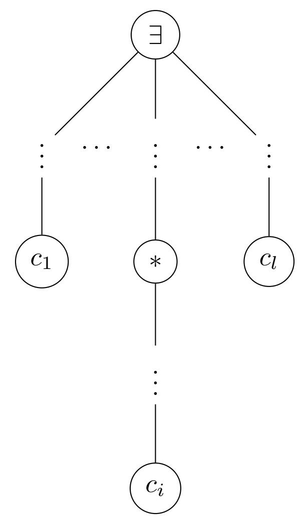

Figure 1: An example PJT with clauses $c_1, \ldots, c_i, \ldots, c_l$ that all contain the same variable, its projection node $\exists$, and a node under consideration $*$.

Theorem 2 ([4], rephrased, 'ADD width is equal to the treewidth of the primal graph')**.** *Given a CNF formula ϕ with a tree decomposition of its primal graph of width w, Algorithm 2 (from [4]) returns a PJT of ϕ of width at most $w+1$.*

Definition 3 ([6], rephrased)**.** A *tree decomposition* of a graph G is a pair (T, χ), where T is a tree and $\chi \colon V(T) \to 2^{V(G)}$ is a labelling function, with the following properties:

- $\bigcup_{t \in V(T)} \chi(t) = V(G)$;
- for every edge $e \in E(G)$, there exists $t \in V(T)$ such that e has both endpoints in $\chi(t)$;
- for all $t, t', t'' \in V(T)$, if t' is on the path between t and t'', then $\chi(t) \cap \chi(t'') \subseteq \chi(t')$.

The *width* of tree decomposition (T, χ) is $\max_{t \in V(T)} |\chi(t)| - 1)$. The *treewidth* of graph G is the smallest w such that G has a tree decomposition of width w.

Definition 4. Let $f, g \colon 2^X \to \mathbb{R}$ be pseudo-Boolean functions. Operations such as addition and multiplication are defined pointwise as

$$(f+g)(Y) \coloneqq f(Y) + g(Y),$$

and

$$(fg)(Y) \coloneqq f(Y)g(Y)$$

for all $Y \subseteq X$.

Definition 5. Let $f \colon 2^X \to \mathbb{R}$ be a pseudo-Boolean function, and $x \in X$. Then $f|_{x=0}, f|_{x=1} \colon 2^X \to \mathbb{R}$ are *restrictions* of f defined as

$$f|_{x=0}(Y) \coloneqq f(Y \setminus \{x\}),$$

and

$$f|_{x=1}(Y) \coloneqq f(Y \cup \{x\})$$

for all $Y \subseteq X$.

Definition 6. Let X be a set. For any $x \in X$, *projection* $\exists_x$ is an endomorphism $\exists_x : \mathbb{R}^{2^X} \to \mathbb{R}^{2^X}$ defined as

$$\exists_x f = f|_{x=1} + f|_{x=0}$$

for any $f : 2^X \to \mathbb{R}$.

Lemma 1. *Let $f, g : 2^X \to \mathbb{R}$ be pseudo-Boolean functions represented by ADDs with n and m nodes, respectively, and $x \in X$. Then $f + g$ and fg can be computed in $\mathcal{O}(mn)$ time, and $\exists_x f$ can be computed in $\mathcal{O}(n^2)$ time.*

Proof. Addition and multiplication are implemented by `apply` algorithm which takes $\mathcal{O}(mn)$ time [1]. Projection consists of two restrictions and an addition by Definition 6. The computational complexity of restriction is dominated by the reduction operation that transforms a decision diagram into a minimal canonical form [1]. While the original reduction algorithm had a $\mathcal{O}(n \log n)$ complexity, caching can reduce it to $\mathcal{O}(n)$ [7]. Either way, the complexity of projection is still $\mathcal{O}(n^2)$. □

Definition 7. Let (T, r, γ, π) be a PJT, and X be the set of variables. The functionality of DPMC execution (i.e., Algorithm 1 in the ADDMC paper [3]) can be represented by $\delta(r)$, where $\delta : \mathcal{V}(T) \to \mathbb{R}^{2^X}$ is a recursive function defined as

$$\delta(t) = \begin{cases} \gamma(t) & \text{if } t \in \mathcal{L}(T) \\ \exists_{\pi(t)} \prod_{u \in \mathcal{C}(t)} \delta(u) & \text{otherwise.} \end{cases}$$

The range of $\delta(r)$ then contains a single real number, i.e., the answer.

Lemma 2. *The number of variables that can appear in the ADDs within a PJT node is $\leq k + 1$.*

Proof. All such variables are introduced in a 'clause' below and projected either in this node or in one of its ancestors. Each such variable would be in the bag of each clause that introduced it. By the tree decomposition properties, it will also be in the bag of both the projection node and the current node because they are on the path between the clause nodes (see Fig. 1).

How do I formalise this argument?

□

Theorem 3. DPMC *is FPT w.r.t. the PJT width. More specifically,* DPMC *running time is $\mathcal{O}(4^k m(n + k))$, where k is the width, n is the number of clauses, and m is the number of variables.*

Proof.

- An ADD with k variables can have up to $2^0 + 2^1 + \cdots + 2^k = 2^{k+1} - 1 = \mathcal{O}(2^k)$ nodes (including leaves).
- Multiplying m ADDs can then take up to $(m-1)(2^{k+1}-1)^2 = \mathcal{O}(m4^k)$ (since one multiplication takes up to $(2^{k+1}-1)^2$ time and the and the result will have up to $2^{k+1} - 1$ nodes because the domain stays the same).
- Each projection consists of two linear operations and a quadratic operation, so projecting m variables will take $\mathcal{O}(4^k m)$ time.
- In total, operations on a PJT node with m ADDs, k variables, n of which are projected takes $\mathcal{O}(4^k(m + n))$ time.
- The total time taken by the DPMC execution stage is the sum of the time taken 'inside' each PJT node. The number of such nodes is upper bounded by m, so the total time complexity of DPMC is $\mathcal{O}(4^k m(n + k))$.

□

References

[1] Bryant, R. E. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers 35*, 8 (1986), 677–691.

[2] Choi, A., Kisa, D., and Darwiche, A. Compiling probabilistic graphical models using sentential decision diagrams. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013. Proceedings* (2013), L. C. van der Gaag, Ed., vol. 7958 of *Lecture Notes in Computer Science*, Springer, pp. 121–132.

[3] Dudek, J. M., Phan, V., and Vardi, M. Y. ADDMC: weighted model counting with algebraic decision diagrams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020* (2020), AAAI Press, pp. 1468–1476.

[4] Dudek, J. M., Phan, V. H. N., and Vardi, M. Y. DPMC: weighted model counting by dynamic programming on project-join trees. In *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings* (2020), H. Simonis, Ed., vol. 12333 of *Lecture Notes in Computer Science*, Springer, pp. 211–230.

[5] Kwisthout, J., Bodlaender, H. L., and van der Gaag, L. C. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings* (2010), H. Coelho, R. Studer, and M. J. Wooldridge, Eds., vol. 215 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 237–242.

[6] Robertson, N., and Seymour, P. D. Graph minors. III. planar tree-width. *J. Comb. Theory, Ser. B 36*, 1 (1984), 49–64.

[7] Somenzi, F. CUDD: CU decision diagram package release 3.0.0. *University of Colorado at Boulder* (2015).